# Story 1.2: PostgreSQL Database Setup & LiquidBase Changesets

Status: drafted

## Story

As a Developer,
I want PostgreSQL 15 configurado con LiquidBase changesets y schema base,
so that Puedo persistir aggregates con garantía de esquema versionado siguiendo estándares corporativos.

## Acceptance Criteria

### AC1: LiquidBase Infrastructure Ready

**Given** El proyecto tiene Spring Boot 3.2+ y Java 21
**When** Agrego las dependencias de LiquidBase Core
**Then**

- `pom.xml` incluye `liquibase-core` (versión gestionada por Spring Boot)
- Estructura de directorios creada: `src/main/resources/liquibase/`
  - `changelog-master.yaml`
  - `changes/dev/`
  - `changes/uat/`
  - `changes/prod/`
- Configuration en `application.yml`:
  - `spring.liquibase.enabled=true`
  - `spring.liquibase.change-log=classpath:liquibase/changelog-master.yaml`
  - `spring.liquibase.contexts=dev` (en `application-local.yml`)

### AC2: PostgreSQL Docker Compose Service

**Given** El proyecto tiene Docker instalado
**When** Creo `docker-compose.yml` en root
**Then**

- Servicio `postgres` configurado con:

    - Imagen: `postgres:15-alpine`
    - Puerto: `5432:5432`
    - Variables de entorno: `POSTGRES_DB=signature_router`, `POSTGRES_USER=siguser`, `POSTGRES_PASSWORD=sigpass`
    - Volume: `postgres-data:/var/lib/postgresql/data`
    - Healthcheck: `pg_isready -U siguser`
- Comando `docker-compose up -d postgres` levanta PostgreSQL exitosamente
- Logs muestran "database system is ready to accept connections"

### AC3: Datasource Configuration & HikariCP Pool

**Given** PostgreSQL running en Docker
**When** Configuro `application-local.yml`
**Then**

- Datasource configurado:
    - `spring.datasource.url=jdbc:postgresql://localhost:5432/signature_router`
    - `spring.datasource.username=siguser`
    - `spring.datasource.password=sigpass` (nota: usar Vault en prod)
    - `spring.datasource.driver-class-name=org.postgresql.Driver`
- HikariCP pool optimizado:
    - `spring.datasource.hikari.maximum-pool-size=20`
    - `spring.datasource.hikari.connection-timeout=2000`
    - `spring.datasource.hikari.idle-timeout=600000`
    - `spring.datasource.hikari.pool-name=SignatureRouterPool`

### AC4: ChangeSet 0001 - UUIDv7 Function

**Given** LiquidBase configurado
**When** Creo `src/main/resources/liquibase/changes/dev/0001-create-uuidv7-function.yaml`
**Then**

- ChangeSet incluye campos obligatorios:
    - `id: "0001"`
    - `author: "BMAD Dev Agent <bmad@signature-router.com>"`

    - `context: dev`
- SQL crea función `uuid_generate_v7()` según spec en `docs/architecture/03-database-schema.md` líneas 133-154
- Bloque `rollback` incluido: `DROP FUNCTION IF EXISTS uuid_generate_v7()`
- Al iniciar app, función existe en PostgreSQL: `SELECT uuid_generate_v7()` retorna UUID válido

#### AC5: ChangeSet 0002 - Table signature_request

**Given** UUIDv7 function creada
**When** Creo `0002-create-signature-request-table.yaml`
**Then**

- Tabla `signature_request` creada con columnas (según `docs/architecture/03-database-schema.md` líneas 26-35):
    - `id UUID PRIMARY KEY DEFAULT uuid_generate_v7()`
    - `customer_id VARCHAR(255) NOT NULL`
    - `transaction_context JSONB NOT NULL`
    - `status VARCHAR(50) NOT NULL`
    - `active_challenge_id UUID NULLABLE`
    - `created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP`
    - `updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP`
    - `expires_at TIMESTAMPTZ NOT NULL`
- Constraint CHECK: `status IN ('PENDING', 'CHALLENGE_SENT', 'COMPLETED', 'FAILED', 'EXPIRED')`
- Índice GIN en `transaction_context`: `CREATE INDEX idx_signature_request_context ON signature_request USING GIN (transaction_context)`
- Índice B-tree en `customer_id`: `CREATE INDEX idx_signature_request_customer ON signature_request (customer_id)`
- Rollback: `DROP TABLE IF EXISTS signature_request CASCADE`

#### AC6: ChangeSet 0003 - Table signature_challenge

**Given** Tabla `signature_request` creada
**When** Creo `0003-create-signature-challenge-table.yaml`
**Then**

- Tabla `signature_challenge` creada con columnas (líneas 42-55):
  - `id UUID PRIMARY KEY DEFAULT uuid_generate_v7()`
  - `signature_request_id UUID NOT NULL REFERENCES signature_request(id) ON DELETE CASCADE`
  - `channel_type VARCHAR(50) NOT NULL`
  - `provider VARCHAR(100) NOT NULL`
  - `provider_challenge_id VARCHAR(255)`
  - `provider_proof TEXT` (cryptographic receipt)
  - `status VARCHAR(50) NOT NULL`
  - `sent_at TIMESTAMPTZ`
  - `responded_at TIMESTAMPTZ`
  - `expires_at TIMESTAMPTZ NOT NULL`
  - `error_code VARCHAR(100)`
  - `raw_response TEXT`
  - `created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP`
- Constraint CHECK: `channel_type IN ('SMS', 'PUSH', 'VOICE', 'BIOMETRIC')`
- Constraint CHECK: `status IN ('PENDING', 'SENT', 'COMPLETED', 'FAILED', 'EXPIRED')`
- FK constraint con ON DELETE CASCADE
- Índices: `idx_challenge_request` (signature_request_id), `idx_challenge_provider` (provider, status)
- Rollback incluido

#### AC7: ChangeSet 0004-0007 - Remaining Tables

**Given** Tablas core creadas
**When** Creo changesets 0004, 0005, 0006, 0007
**Then**

- `0004-create-routing-rule-table.yaml`: Tabla `routing_rule` (líneas 60-71)
  - Columnas: id, name UNIQUE, description, priority, condition (SpEL), target_channel, enabled, created_at, updated_at, created_by, updated_by
  - UNIQUE constraint en `name`
  - Índice en `priority, enabled`
- `0005-create-connector-config-table.yaml`: Tabla `connector_config` (líneas 75-87)

    - Columnas: id, provider UNIQUE, enabled, config JSONB, vault_path, degraded_mode, degraded_since, error_rate, last_health_check, created_at, updated_at
    - Índice GIN en `config`
- `0006-create-outbox-event-table.yaml`: Tabla `outbox_event` (líneas 92-100)
    - Columnas: id, aggregate_id, aggregate_type, event_type, payload JSONB, payload_hash, created_at, published_at
    - Índice en `published_at` (para Debezium)
    - Índice en `aggregate_id`
- `0007-create-audit-log-table.yaml`: Tabla `audit_log` (líneas 104-115)
    - Columnas: id, entity_type, entity_id, action, user_id, ip_address, changes JSONB, created_at
    - Índice en `created_at` (para rotación)
    - Índice en `entity_type, entity_id`
- Todos incluyen rollback obligatorio

#### AC8: Changelog Master Configuration

**Given** Los 7 changesets creados en `changes/dev/`
**When** Configuro `changelog-master.yaml`
**Then**

- Contenido YAML:

```yaml
databaseChangeLog:
  - includeAll:
      path: changes/dev
      relativeToChangelogFile: true
  - includeAll:
      path: changes/uat
      relativeToChangelogFile: true
  - includeAll:
      path: changes/prod
      relativeToChangelogFile: true
```

- LiquidBase ejecuta changesets en orden alfabético (0001, 0002, ... 0007)
- Changesets uat/prod inicialmente vacíos (se copiarán de dev cuando se promueva)

### AC9: LiquidBase Execution on Startup

**Given** Todos los changesets configurados
**When** Inicio la aplicación con `./mvnw spring-boot:run -Dspring.profiles.active=local`
**Then**

- Logs muestran:
  - `Liquibase: Reading from liquibase.changelog-master.yaml`
  - `Liquibase: Successfully acquired change log lock`
  - `Liquibase: Creating database history table [PUBLIC.DATABASECHANGELOG]`
  - `Liquibase: Running Changeset: changes/dev/0001-create-uuidv7-function.yaml::0001::BMAD Dev Agent`
  - `Liquibase: Running Changeset: changes/dev/0002-create-signature-request-table.yaml::0002::BMAD Dev Agent`
  - ... (hasta 0007)
  - `Liquibase: Successfully released change log lock`
- Tabla `DATABASECHANGELOG` contiene 7 registros (uno por changeset)
- Tabla `DATABASECHANGELOGLOCK` existe con lock released

### AC10: Schema Validation with psql

**Given** LiquidBase ejecutó exitosamente
**When** Ejecuto `docker exec -it <postgres-container> psql -U siguser -d signature_router -c "\dt"`
**Then**

- Lista de tablas muestra:
  - `signature_request`
  - `signature_challenge`
  - `routing_rule`
  - `connector_config`
  - `outbox_event`
  - `audit_log`
  - `databasechangelog`
  - `databasechangeloglock`

**And** Comando `\d signature_request` muestra columnas correctas con tipos UUIDv7, JSONB, constraints

### AC11: Integration Test with Testcontainers

**Given** Changesets listos
**When** Creo test de integración `DatabaseSchemaIntegrationTest.java`
**Then**

- Test usa `@Testcontainers` con PostgreSQL 15
- Verifica que las 6 tablas de negocio existen
- Verifica que `uuid_generate_v7()` function existe
- Inserta un registro en `signature_request` con ID generado por función UUIDv7
- Query recupera el registro exitosamente
- Test pasa en `mvn verify`

### AC12: Rollback Test (Manual Validation)

**Given** Schema creado con LiquidBase
**When** Ejecuto `./mvnw liquibase:rollback -Dliquibase.rollbackCount=1`
**Then**

- Última tabla (audit_log) es eliminada
- Log muestra: `Rolling Back Changeset: changes/dev/0007-create-audit-log-table.yaml::0007::BMAD Dev Agent`
- Tabla `DATABASECHANGELOG` ahora tiene 6 registros (en lugar de 7)
- Re-ejecutar app vuelve a crear tabla `audit_log`

## Tasks / Subtasks

### Task 1: Configure LiquidBase Dependencies & Directory Structure (AC: #1)

- ◯ 1.1. Agregar dependency `liquibase-core` a `pom.xml` (versión gestionada por Spring Boot 3.2+)
- ◯ 1.2. Crear estructura de directorios:
  - `src/main/resources/liquibase/changes/dev/`
  - `src/main/resources/liquibase/changes/uat/`
  - `src/main/resources/liquibase/changes/prod/`
- ◯ 1.3. Crear `src/main/resources/liquibase/changelog-master.yaml` con `includeAll` paths

◯ 1.4. Configurar `application.yml`:

- `spring.liquibase.enabled: true`
- `spring.liquibase.change-log: classpath:liquibase/changelog-master.yaml`

◯ 1.5. Configurar `application-local.yml`:

- `spring.liquibase.contexts: dev`

### Task 2: Create Docker Compose for PostgreSQL 15 (AC: #2)

◯ 2.1. Crear `docker-compose.yml` en root con servicio `postgres`:

- Imagen: `postgres:15-alpine`
- Puerto: 5432
- Variables: POSTGRES_DB, POSTGRES_USER, POSTGRES_PASSWORD
- Volume: `postgres-data`
- Healthcheck: `pg_isready -U siguser`

◯ 2.2. Agregar `docker-compose.yml` a `.gitignore` (opcional: usar `.env` para secrets)

◯ 2.3. Documentar comandos en README.md:

- `docker-compose up -d postgres`
- `docker-compose logs -f postgres`
- `docker-compose down -v` (eliminar volumen)

◯ 2.4. Verificar que PostgreSQL levanta exitosamente:

- `docker exec -it <container> psql -U siguser -d signature_router -c "SELECT version();"`

### Task 3: Configure Spring Datasource & HikariCP Pool (AC: #3)

◯ 3.1. Agregar dependency `postgresql` driver a `pom.xml`

◯ 3.2. Configurar `application-local.yml`:

- `spring.datasource.url: jdbc:postgresql://localhost:5432/signature_router`
- `spring.datasource.username: siguser`
- `spring.datasource.password: sigpass`
- `spring.datasource.driver-class-name: org.postgresql.Driver`

◯ 3.3. Configurar HikariCP pool (en `application-local.yml`):

- `spring.datasource.hikari.maximum-pool-size: 20`
- `spring.datasource.hikari.connection-timeout: 2000`
- `spring.datasource.hikari.idle-timeout: 600000`
- `spring.datasource.hikari.pool-name: SignatureRouterPool`

◯ 3.4. Agregar comentario en `application.yml` indicando usar Vault en prod

◯ 3.5. Verificar conexión en startup logs: `HikariPool-1 - Start completed`

### Task 4: Create ChangeSet 0001 - UUIDv7 Function (AC: #4)

◯ 4.1. Crear archivo `src/main/resources/liquibase/changes/dev/0001-create-uuidv7-function.yaml`

◯ 4.2. Definir changeset con:

- `id: "0001"`
- `author: "BMAD Dev Agent <bmad@signature-router.com>"`
- `context: dev`

◯ 4.3. Agregar SQL para función `uuid_generate_v7()` (según `docs/architecture/03-database-schema.md` líneas 133-154)

◯ 4.4. Agregar bloque `rollback`:

- `sql: "DROP FUNCTION IF EXISTS uuid_generate_v7()"`

◯ 4.5. Copiar archivo a `changes/uat/` y `changes/prod/` con `context: uat` y `context: prod`

◯ 4.6. Validar YAML syntax con parser online

◯ 4.7. Test manual: iniciar app y verificar función existe en PostgreSQL

### Task 5: Create ChangeSet 0002 - Table signature_request (AC: #5)

◯ 5.1. Crear archivo `0002-create-signature-request-table.yaml`

◯ 5.2. Definir changeset (id: 0002, author, context: dev)

◯ 5.3. Agregar `createTable` con columnas:

- `id` (UUID, PK, default: `uuid_generate_v7()`)
- `customer_id` (VARCHAR 255, NOT NULL)
- `transaction_context` (JSONB, NOT NULL)
- `status` (VARCHAR 50, NOT NULL)
- `active_challenge_id` (UUID, NULLABLE)

- `created_at`, `updated_at`, `expires_at` (TIMESTAMPTZ)

◯ 5.4. Agregar constraint CHECK en `status`:

- Valores permitidos: PENDING, CHALLENGE_SENT, COMPLETED, FAILED, EXPIRED

◯ 5.5. Crear índice GIN en `transaction_context`:

- `createIndex` → `idx_signature_request_context` → USING GIN

◯ 5.6. Crear índice B-tree en `customer_id`

◯ 5.7. Agregar rollback: `dropTable: signature_request`

◯ 5.8. Copiar a uat/prod con context adecuado

◯ 5.9. Validar schema con `\d signature_request` en psql

### Task 6: Create ChangeSet 0003 - Table signature_challenge (AC: #6)

◯ 6.1. Crear archivo `0003-create-signature-challenge-table.yaml`

◯ 6.2. Definir changeset (id: 0003, author, context: dev)

◯ 6.3. Agregar `createTable` con 13 columnas (según líneas 42-55 del schema doc):

- `id`, `signature_request_id` (FK a signature_request), `channel_type`, `provider`, etc.

◯ 6.4. Agregar FK constraint:

- `addForeignKeyConstraint` → `fk_challenge_request` → referencia `signature_request(id)` → `onDelete: CASCADE`

◯ 6.5. Agregar constraint CHECK en `channel_type`:

- Valores: SMS, PUSH, VOICE, BIOMETRIC

◯ 6.6. Agregar constraint CHECK en `status`:

- Valores: PENDING, SENT, COMPLETED, FAILED, EXPIRED

◯ 6.7. Crear índices:

- `idx_challenge_request` en `signature_request_id`
- `idx_challenge_provider` en `(provider, status)`

◯ 6.8. Agregar rollback completo

◯ 6.9. Copiar a uat/prod

### Task 7: Create ChangeSets 0004-0007 for Remaining Tables (AC: #7)

◯ 7.1. **ChangeSet 0004** - `routing_rule`:

- Columnas: id, name (UNIQUE), description, priority, condition, target_channel, enabled, created_at, updated_at, created_by, updated_by

- UNIQUE constraint en `name`
- Índice en `(priority, enabled)`
- Rollback

◯ 7.2. **ChangeSet 0005** - `connector_config`:
- Columnas: id, provider (UNIQUE), enabled, config (JSONB), vault_path, degraded_mode, degraded_since, error_rate, last_health_check, created_at, updated_at
- Índice GIN en `config`
- Rollback

◯ 7.3. **ChangeSet 0006** - `outbox_event`:
- Columnas: id, aggregate_id, aggregate_type, event_type, payload (JSONB), payload_hash, created_at, published_at
- Índice en `published_at` (crítico para Debezium CDC)
- Índice en `aggregate_id`
- Rollback

◯ 7.4. **ChangeSet 0007** - `audit_log`:
- Columnas: id, entity_type, entity_id, action, user_id, ip_address, changes (JSONB), created_at
- Índice en `created_at` (para rotación de logs)
- Índice en `(entity_type, entity_id)`
- Rollback

◯ 7.5. Copiar todos a uat/prod

### Task 8: Validate LiquidBase Execution on Startup (AC: #9)

◯ 8.1. Iniciar app con perfil `local`: `./mvnw spring-boot:run -Dspring.profiles.active=local`

◯ 8.2. Verificar logs de LiquidBase:
- "Liquibase: Reading from liquibase.changelog-master.yaml"
- "Running Changeset: changes/dev/0001-create-uuidv7-function.yaml::0001::BMAD Dev Agent"
- ... (hasta 0007)
- "Successfully released change log lock"

◯ 8.3. Verificar tabla `DATABASECHANGELOG` tiene 7 registros:

- `docker exec -it <container> psql -U siguser -d signature_router -c "SELECT id, author, filename FROM databasechangelog ORDER BY orderexecuted;"`

◯ 8.4. Verificar que re-iniciar app NO ejecuta changesets de nuevo (idempotencia)

### Task 9: Create Integration Test with Testcontainers (AC: #11)

◯ 9.1. Agregar dependencies a `pom.xml`:
- `testcontainers-junit-jupiter`
- `testcontainers-postgresql`

◯ 9.2. Crear test `src/test/java/com/bank/signature/infrastructure/DatabaseSchemaIntegrationTest.java`

◯ 9.3. Configurar `@Testcontainers` con PostgreSQL 15 container

◯ 9.4. Escribir test methods:
- `testAllTablesExist()` → verifica 6 tablas de negocio + 2 de LiquidBase
- `testUuidV7FunctionExists()` → ejecuta `SELECT uuid_generate_v7()`
- `testInsertSignatureRequest()` → inserta registro con UUIDv7 auto-generado
- `testJsonbColumnWorks()` → inserta/recupera JSONB en `transaction_context`

◯ 9.5. Ejecutar test: `./mvnw verify`

◯ 9.6. Verificar que test pasa en CI/CD pipeline

### Task 10: Document Schema & LiquidBase Workflow (AC: #12)

◯ 10.1. Actualizar README.md con sección "Database Setup":
- Comandos Docker Compose
- Comandos LiquidBase (`./mvnw liquibase:status`, `liquibase:rollback`)
- Estructura de changesets

◯ 10.2. Crear `docs/development/database-migrations.md` con:
- Estándares de LiquidBase (YAML format, contexts, rollback)
- Naming conventions (0001, 0002, etc.)
- Flujo de promoción (dev → uat → prod)
- Testing strategy con Testcontainers

◯ 10.3. Agregar comentarios en cada changeset YAML explicando propósito

◯ 10.4. Test manual de rollback:

- `./mvnw liquibase:rollback -Dliquibase.rollbackCount=1`
- Verificar que tabla `audit_log` se elimina
- Re-iniciar app y verificar que se recrea

## Dev Notes

### Architecture Patterns & Constraints

- **Hexagonal Architecture**: Este story está en la capa de **Infrastructure** (database setup)
- **DDD Alignment**: Las tablas corresponden a aggregates (`signature_request`, `signature_challenge`) y domain events (`outbox_event`)
- **Outbox Pattern**: Tabla `outbox_event` es crítica para garantizar atomicidad (estado + evento en misma TX)
- **UUIDv7**: Sortable UUIDs para mejor performance en índices B-tree (vs UUIDv4 random)
- **JSONB**: `transaction_context` y `config` usan JSONB para flexibilidad sin comprometer queries (GIN indexes)
- **Non-repudiation**: `provider_proof` en `signature_challenge` almacena receipt criptográfico del provider
- **Pseudonymization**: `customer_id` NO es PII directo (hashed/tokenized), cumple GDPR

### Source Tree Components to Touch

```
signature-router/
├── pom.xml                                    # [MODIFY] agregar liquibase-core,
postgresql driver, testcontainers
├── docker-compose.yml                         # [CREATE] servicio postgres
├── src/main/resources/
│   ├── application.yml                        # [MODIFY] spring.liquibase config
base
│   ├── application-local.yml                  # [MODIFY] datasource + hikari
pool config
│   └── liquibase/
│       ├── changelog-master.yaml              # [CREATE] master changelog con
includeAll
│       └── changes/
│           ├── dev/
│           │   ├── 0001-create-uuidv7-function.yaml           # [CREATE]
│           │   ├── 0002-create-signature-request-table.yaml   # [CREATE]
│           │   ├── 0003-create-signature-challenge-table.yaml # [CREATE]
│           │   ├── 0004-create-routing-rule-table.yaml        # [CREATE]
│           │   ├── 0005-create-connector-config-table.yaml    # [CREATE]
│           │   ├── 0006-create-outbox-event-table.yaml        # [CREATE]
```

```
│           │   └── 0007-create-audit-log-table.yaml           # [CREATE]
│           ├── uat/                                 # [CREATE] copiar changesets con
context: uat
│           └── prod/                                # [CREATE] copiar changesets con
context: prod
├── src/test/java/com/bank/signature/infrastructure/
│   └── DatabaseSchemaIntegrationTest.java      # [CREATE] Testcontainers test
└── docs/development/
    └── database-migrations.md                  # [CREATE] LiquidBase workflow
documentation
```

### Testing Standards Summary

- **Unit Tests**: No aplicable (schema creation es responsabilidad de LiquidBase)
- **Integration Tests**:
  - `DatabaseSchemaIntegrationTest.java` con `@Testcontainers`
  - Verifica schema completo, constraints, índices, función UUIDv7
  - Debe pasar en `mvn verify` antes de merge
- **Manual Tests**:
  - Rollback test con `liquibase:rollback -Dliquibase.rollbackCount=1`
  - Validar idempotencia (re-ejecutar changesets no falla)
- **CI/CD Pipeline**:
  - Docker Compose up en pipeline
  - Ejecutar `mvn verify` (incluye Testcontainers tests)
  - Validar que DATABASECHANGELOG tiene exactamente 7 registros

### Project Structure Notes

- **LiquidBase Directory Structure**: Alineado con estándar corporativo (changes/{dev,uat,prod})
- **ChangeSet Naming**: Prefijo numérico (0001, 0002, etc.) garantiza orden alfabético = orden de ejecución
- **YAML Format**: Preferido sobre XML/SQL por legibilidad y rollback declarativo
- **Contexts**: Uso de `context: dev/uat/prod` permite ejecutar changesets específicos por entorno
- **Rollback Blocks**: **MANDATORY** en cada changeset (corporate standard)

### References

- **[Source: docs/architecture/03-database-schema.md]**: Schema completo con DDL para las 6 tablas
  - Líneas 26-35: `signature_request` table definition
  - Líneas 42-55: `signature_challenge` table definition
  - Líneas 60-71: `routing_rule` table definition
  - Líneas 75-87: `connector_config` table definition
  - Líneas 92-100: `outbox_event` table definition
  - Líneas 104-115: `audit_log` table definition
  - Líneas 133-154: UUIDv7 function implementation (PostgreSQL 15+)
- **[Source: docs/sprint-artifacts/tech-spec-epic-1.md]**: LiquidBase Migration Strategy
  - Líneas 176-221: Directory structure, changelog-master.yaml, changeset standards
  - Líneas 223-290: Example changeset format (0002-create-signature-request-table.yaml)
- **[Source: docs/epics.md]**: Story 1.2 definition
  - Líneas 175-208: Acceptance Criteria original, Technical Notes
- **[Source: docs/prd.md]**: Database requirements
  - Functional requirements FR1-FR46 mapping to tables
  - Non-functional requirements: performance, compliance, audit

### Corporate LiquidBase Standards (Critical)

- **Directory Structure**: `liquibase/changes/{dev,uat,prod}/`
- **Naming Convention**: `0001-descriptive-name.yaml`, `0002-another-table.yaml`, etc.
- **Mandatory Fields**: `id`, `author`, `context`, `changes`, `rollback`
- **YAML Format**: Preferred over XML/SQL
- **Contexts**: Explicit context per environment (`dev`, `uat`, `prod`)
- **Rollback**: **MANDATORY** in every changeset (especially critical in prod)
- **Changelog Master**: Single `changelog-master.yaml` with `includeAll` for each environment directory
- **Idempotency**: LiquidBase manages via DATABASECHANGELOG table (don't execute twice)

## Definition of Done

- ◯ **Code Complete**:
  - ✅ `pom.xml` incluye `liquibase-core` y `postgresql` driver
  - ◯ 7 changesets YAML creados en `liquibase/changes/dev/` (0001-0007)
  - ◯ Changesets copiados a `uat/` y `prod/` con contexts correctos
  - ◯ `changelog-master.yaml` configurado con `includeAll`
  - ◯ `application.yml` y `application-local.yml` configurados
  - ◯ `docker-compose.yml` con servicio PostgreSQL 15
  - ◯ `DatabaseSchemaIntegrationTest.java` creado y passing
- ◯ **Tests Passing**:
  - ◯ Integration test con Testcontainers pasa en `mvn verify`
  - ◯ Manual test: `docker-compose up -d postgres` + `mvn spring-boot:run` ejecuta changesets exitosamente
  - ◯ Manual test: `psql` verifica 6 tablas creadas con schema correcto
  - ◯ Manual test: rollback elimina última tabla y re-ejecutar la recrea
- ◯ **Architecture Validated**:
  - ◯ ChangeSet structure respeta estándares corporativos (YAML, contexts, rollback)
  - ◯ Tabla `outbox_event` lista para Debezium CDC (Story 1.3)
  - ◯ UUIDv7 function validada con test unitario (genera UUID sortable)
  - ◯ FK constraints configuradas con `ON DELETE CASCADE` donde corresponde
- ◯ **Documentation Updated**:
  - ◯ README.md incluye sección "Database Setup" con comandos Docker Compose
  - ◯ `docs/development/database-migrations.md` creado con workflow LiquidBase
  - ◯ Cada changeset incluye comentarios explicando propósito
  - ◯ CHANGELOG.md actualizado: "Added PostgreSQL 15 setup with LiquidBase changesets"
- ◯ **Code Review Approved**:
  - ◯ Peer review confirma changesets YAML syntax correcta
  - ◯ Verificar rollback blocks en todos los changesets
  - ◯ Validar que contexts están correctamente asignados (dev/uat/prod)
  - ◯ Confirmar que índices GIN/B-tree están optimizados según schema doc
- ◯ **Story Marked as Done**:

- ◯ Todos los 12 Acceptance Criteria verificados ✅
- ◯ Sprint status actualizado: `1-2-postgresql-database-setup-liquidbase-changesets: done`
- ◯ Story list actualizada en `docs/sprint-artifacts/sprint-status.yaml`

---

## Dev Agent Record

### Context Reference

### Agent Model Used

Claude Sonnet 4.5

### Debug Log References

### Completion Notes List

### File List

**Created:**

**Modified:**

**Deleted:**

---

## Change Log

| Date | Author | Change |
|---|---|---|
| 2025-11-26 | BMAD SM Agent | Story 1.2 draft created with LiquidBase standards |